VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2012 Tomáš Trval

VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA INFORMATIKY

Virtuální vegetace

Virtual Vegetation

2012 Tomáš Trval

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Trval**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Virtuální vegetace
Virtual Vegetation

Zásady pro vypracování:

Virtuální vegetace se využívá jak v architektuře, tak i v počítačových hrách a umění. Úkolem této práce je příprava systému pro generování vegetace. Tento systém by měl být ve formě aplikace, nebo knihovny, která bude schopna generovat modely stromů a keřů, případně i travin. Cílem práce není vytvořit vzhledově dokonalou vegetaci, ale pochopit, popsat a implementovat základní mechanizmy generování.

1. Student provede průzkum již existujících nastrojů a metod pro generování vegetace.
2. Kriticky tyto nástroje a metody porovná.
3. Provede návrh a implementaci systému generování vegetace.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Karel Mozdřeň**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Podpis: .....

Abstrakt

Tato bakalářská práce se zabývá problematikou generováním virtuálních rostlin. V úvodu vysvětluji motivaci k vytvoření této práce a představuji problémy, kterými se bude zabývat. Teoretická část shrnuje historický vývoj metod užívaných pro tvorbu virtuální vegetace. Dále hodnotím tři aplikace zabývající se touto problematikou. Další kapitola je hlavní částí práce. Popisuje návrh a implementaci modulu VegGen určeného pro generování stromů a keřů. Následuje prezentace experimentů a výsledků vytvořeného modulu. V závěru práce jsou shrnuty získané zkušenosti a je nastíněno budoucí využití modulu a jeho částí.

Klíčová slova: *L-Systémy, Procedurální Rostliny, Virtuální Vegetace, Počítačová Grafika, Blender*

Abstract

This bachelor thesis deals with the problem of generating the virtual plants. In the introduction I explain my motivation to create this work and I present problems to be addressed. The theoretical section summarizes the historical development of methods used for the creation of virtual vegetation. Furthermore I reviewed three applications which deals this problem. The next chapter is the main part of the work. It describes the proposal and implementation of the module VegGen. The module is designed for generating trees and shrubs. The presentation of experiments and the results of developed module is the next part of the thesis. The conclusion summarizes the experience gained and work itself. Also, conclusion outlines the future use of the module and its parts.

Keywords: *L-systems, Procedural Plants, Virtual Vegetation, Computer Graphics, Blender*

Seznam použitých zkratek a symbolů

3D	trojrozměrný
DNA	deoxyribonukleová kyselina
L-systém	Lindenmayerův systém

Obsah

1 Úvod

Člověk v dnešní době stále častěji naráží na virtuální realitu. Vidí ji v televizi i v počítači. Dokonce na něj vystupuje i z pláten kin. Tyto uměle vytvořené světy jsou často pouze obrazem světa skutečného. I když se tomu mnohdy nechce věřit. Většina i těch nejbizarnějších bytostí, obrovských myslících strojů nebo podivuhodných rostlin je inspirována organismy ze Země. Jejich autoři čerpají své nápady z oblastí, s kterými přijde běžný člověk málokdy do styku. Vděčnými múzami pro složitě vypadající mechanické zázraky bývá říše tropického hmyzu. Pro tvorbu rostlin či celých krajin, které vypadají jako z jiné planety, bývá předlohou podmořský svět.

Obrázek 1: Příklad využití virtuální vegetace, zdroj: [9]

Snad každého někdy fascinoval svět zvláštních efektů a nádherných virtuálních scenérií. V poledních letech se staly pro běžné smrtelníky dostupnými počítače umožňující vytvářet a vizualizovat vlastní virtuální objekty v rozumném čase. Proto jsem se po zahájení studia na vysoké škole začal zajímat o tvorbu 3D grafiky. Nejvíce mne zaujaly přírodní scenérie. Bohužel pro mne bylo obtížné vytvořit reálně vypadající rostlinu natož celou scénu. Časem jsem si však uvědomil jednu zajímavou skutečnost. Při modelování některých objektů je vhodné použít stejné postupy, jako při jejich tvorbě ve skutečném světě. První takovou zkušeností pro mne byla tvorba kontrabasu. Nejprve je třeba vytvořit jednotlivé dřevěné díly. Použít na ně správné materiály. Poté je přimknout k sobě. Dodat kolíky, kobylku a struník pro ladění a nakonec navinout samotné struny. Najednou mi došlo, že pokud chci tvořit rostliny musím také zopakovat proces jejich růstu.

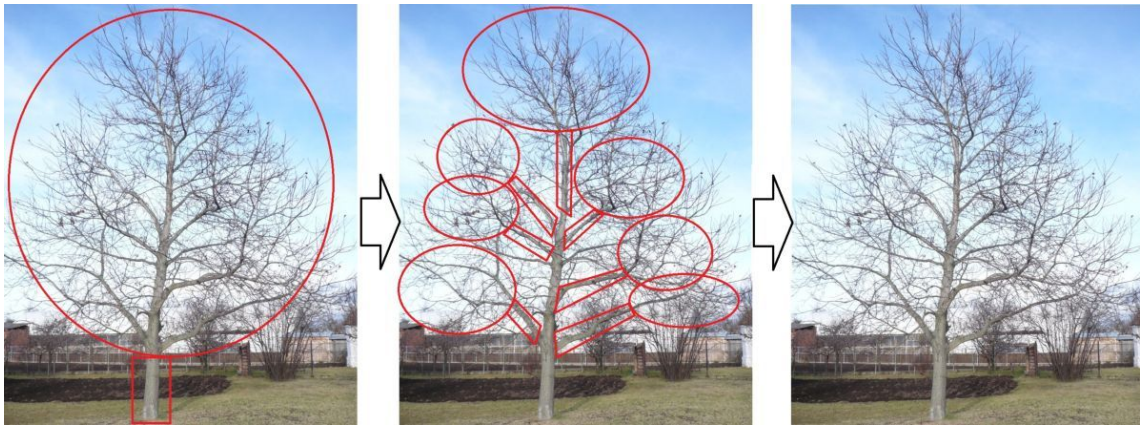
Ve své bakalářské práci se zabývám tvorbou virtuální vegetace. Nejprve se snažím shrnout vývoj metod používaných pro generování rostlin. Stejně jako vývoj jejich praktického využití u projektů požadujících zobrazení scény v reálném čase. Některé nástroje jsem též osobně vyzkoušel a zhodnotil jejich použitelnost a vstřícnost k uživateli. Jednalo se o dva zástupce z nezávislé tvorby. Oba nástroje jsou dostupné zdarma. A pak o komerční produkt, který má za sebou již devítiletou historii. Mohl jsem u něj tedy provést porovnání starších a novějších verzí.

Cílem této práce je realizace vlastního modulu pro generování vegetace. Modul bude do scény vkládat jednotlivé modely stromů a keřů. Bude generovat jedince různých tvarů avšak tak, aby odpovídali svou podobou druhu rostliny nastavenému pomocí parametrů. Za tímto účelem poskytne modul uživateli možnost ovlivňovat proces tvorby rostliny, nejlépe skrze uživatelské rozhraní. Cílem práce není vytvořit vzhledově zcela dokonalou vegetaci. Důraz bude kladen především na pochopení a implementaci základních mechanizmů generování rostlin.

Ke konci této práce budou provedeny experimenty. Jejich cílem bude vygenerovat pomocí vytvořeného modulu stromy a keře dle fotografií jejich protějšků z přírody. V hlavní části experimentů tedy budou prezentovány dvojce snímků vzorových a vygenerovaných rostlin. V poslední části experimentů budou slovně zhodnoceny jejich vizuální rozdíly. V sekci závěr budou shrnuty dosažené výsledky a nově získané zkušenosti. Nastíním též moji představu o budoucím vývoji modulu a jeho částí.

2 State of The Art

Při pozorování stromu se na výsledný vjem projevuje míra lidské pozornosti. Pokud rostlina není hlavním objektem ve scéně, je většinou vnímána jako kmen a koruna (Obr. 2). Když začne strom upoutávat zrak pozorovatele více, jsou kolem kmenu patrné hlavní větve. Ty opět zakončuje malá koruna. Tímto způsobem lze stále zostřovat míru vnímaných detailů. Samozřejmě jen do té úrovně v níž je strom složen pouze z větví. Porovnáním těchto úrovní vjemů jsem zjistil zajímavou skutečnost. V samotném stromu jsou rozpoznatelné obrazy menších stromů. Také čím podrobněji strom zkoumám, tím je celková délka větví větší. Tyto vztahy mne přivedly k fraktálům.



Obrázek 2: Vývoj vnímání a zpřesňování tvaru stromu

2.1 Fraktály

Fraktál je útvar jehož topologická dimenze je ostře větší než Hausdorfova dimenze. Hausdorfova dimenze udává mírů nepravidelnosti útvaru [3]. Struktura fraktálu vzniká opakováním stejných pravidel stále dokola. Díky tomu lze získat velice složitou strukturu na základě jednoduchých pravidel. Tento princip využívá i příroda. Na základě malého množství informace, tedy jednoduché DNA, lze za pomoci buňky, která provádí předpis DNA, vytvářet opakováním předpisu složité struktury. Až vznikají například celé cévní systémy.

(a) Dopravních komunikace, Moskva (b) Velký barierový útes, Austrálie

Obrázek 3: Fraktály stvořené masami, zdroj aplikace [10]

Kombinací několika systémů mohou vznikat složité organismy jako rostliny a zvířata či člověk. Zajímavé je, že velké množství výsledných jedinců může znovu tvořit rozlehlé fraktály (Obr. 3). Fraktály nemusí tvořit jen živé organismy, ale i pouhá masa částic, které mají stejné chování. Proto můžeme do fraktálů zařadit i tvary delt řek (Obr. 4), či opakující se vlny dun v písečných pouštích. Fraktály zabývající se generováním rostlin se řadí do skupiny Lindenmayerových simulátorů růstu.

Obrázek 4: Delta řeky, zdroj aplikace [10]

.

2.2 Lindenmayerovi simulátory růstu

Lindenmayerovi simulátory růstu (dále jen L-systémy) byly vytvořeny biologem jménem Aristid Lindenmayer. Ten se zabýval množením a růstem bakterii a řas. Aby mohl popsat proces růstu vytvořil svou odnož formálního jazyka (Def. 1) [5].

$$\text{Formální jazyk } L \text{ nad abecedou } \varepsilon \text{ je podmnožinou } \varepsilon^*, \text{ která je množinou slov nad touto abecedou.} \quad (1)$$

Lindenmayerem definovaný jazyk byl postupně vylepšován, aby mohl zachytit většinu procesů růstu a díky tomu vznikaly specializace, které také zahrnujeme pod pojem L-systémy [6].

2.2.1 Deterministické L-systémy

Tyto systémy byly vytvořeny přímo A. Lindenmayerem. Jsou také označovány jako D0L-systémy. Obsahují abecedu, axiom a pravidla. Abeceda A je konečná množina symbolů. Axiom je počáteční posloupnost symbolů od, které se začínají generovat ostatní slova. Pravidlo představuje uspořádanou dvojici (x, y), která se zapisuje ve tvaru $x \to y$, přičemž $x \in A$, $y \in A^*$ [6].

2.2.2 Závorkové L-systémy

Základní D0L-systémy umožňují vytvářet pouze posloupnosti buněk, bez možnosti rozvětvení struktury. Do systému pravidel bylo potřeba přidat prvek reprezentující paměť. Proto A. Lindenmayer zavedl do své abecedy sytém závorek. Paměť je implementována jako zásobník. Sytém závorek umožňuje provádět skoky ve struktuře a označit jednotlivé větve. Levá závorka pro začátek větve, tedy uložení pozice na zásobník a pravá závorka pro konec větve, tedy načtení pozice ze zásobníku. V mé práci se závorkové systémy odráží v použití stromové struktury dat, díky které je možné provádět větvení.

$$\begin{aligned} Strom &= \{N, T, P, S\} \\ N &= \{U\} \\ T &= \{v, +, -\} \\ P &= \{S \to U, U \to v\,[+U]\,[-U]\} \end{aligned} \tag{2}$$

Proces přepisování:

0. generace : S
1. generace : U
2. generace : $v\,[+U]\,[-U]$
3. generace : $v\,[+v\,[+U]\,[-U]]\,[-v\,[+U]\,[-U]]$
4. generace : $v\,[+v\,[+v\,[+U]\,[-U]]\,[-v\,[+U]\,[-U]]]\,[-v\,[+v\,[+U]\,[-U]]\,[-v\,[+U]\,[-U]]]$

(a) 0. generace (b) 1. generace (c) 2. generace (d) 3. generace

Obrázek 5: Růst stromu

Uvedený příklad závorkového deterministického L-systému (2) představuje předpis tvorby jednoduchého stromu. Gramatika je tvořena jednoprvkovou množinou neterminálů

N, kde U reprezentuje větev, která bude ještě růst. Množinou terminálů T, kde v zastupuje nerostoucí větev, $+$ je pokyn pro kreslíře otoč se o úhel $+\alpha$, oproti tomu $-$ je otočení o úhel $-\alpha$. Závorky také představují pokyny pro kreslíře. Levá [říká, zapamatuj si tuto pozici a nasměrování, pravá] znamená vyzvedni a smaž z paměti nejnovější záznam. Načež se přesuň a natoč podle pozice a směru vyzvednutého záznamu. Množina P obsahuje startovní přepisovací pravidlo S a pravidlo pro růst části rostliny N. Pro člověka je nepohodlné vytvářet si na základě vygenerovaného kódu představu stromu. Proto jsem vytvořil tyto obrázky jednotlivých generací stromu (Obr 5). Pro vysvětlení grafickou reprezentací nulté iterace je semínko tedy tečka.

2.2.3 Stochastické L-systémy

Struktury vznikající za pomoci výše popsaných L-systémů, se poněkud liší oproti, těm které lze nalézt v reálném světě. Což je způsobeno tím, že na rostlinu během jejího vývoje neustále působí její okolí a stejně tak ona působí na ně. Tyto změny by šlo provádět složitými simulacemi, které by ovšem předpokládali, že nejdříve vytvořím simulátor okolního prostředí. Toto zahrnuje reálně se chovající fyzikální model, simulace kapalin, sluneční svit a jiné. Naplním jej mikroorganismy, rostlinami, půdou, vodou a živočichy. Všem zadám jejich chování a poté na superpočítači spustím mou simulaci. Pokud je mým cílem vytvořit jedince a ne celý ekosystém, bude přijatelnější simulovat vlivy prostředí pomocí pseudonáhodných změn procesu růstu.

Náhodné změny zavádějí právě stochastické L-systémy. Díky nim lze využít nahodilost s určitou mírou kontroly. Pro zjednodušení budu uvažovat, že L-systém obsahuje pouze jedno pravidlo. Aby se docílilo jisté míry nahodilosti, zavedu pro stejnou levou stranu pravidla více pravých stran. Vzniká tedy více pravidel, z nich má každé svou vlastní váhu. Váha určuje pravděpodobnost s jakou bude konkrétní pravidlo vybráno. V následujících příkladech čísla u šipek reprezentují váhu pravidla. V rostlině by tato pravidla mohla znamenat, že může zachovat svou délku nebo ji zdvojnásobit, či dokonce ztrojnásobit. Což by mohlo vyjadřovat ovlivnění rychlosti růstu nepravidelnostmi srážek. Pro jednoduchost je a jediným symbolem abecedy.

$$\begin{aligned} a &\rightarrow^1 a \\ a &\rightarrow^4 aa \\ a &\rightarrow^2 aaa \end{aligned} \tag{3}$$

Pravděpodobnost, že bude pravidlo p_j vybráno, lze spočíst pro levou stranu a za pomoci vzorce (4). Kde n je počet přepisovacích pravidel pro a, $f(a_i)$ je váha i-tého pravidla.

$$p_j = \frac{f(a_j)}{\sum_{i=1}^{n} f(a_i)} \tag{4}$$

2.2.4 Kontextové L-systémy

Aby bylo možné simulovat reakci rostlin na podměty a zásahy zvenčí, byl do L-systému zařazen kontext. Díky němu dokáže rostlina posílat signály svým tělem. Těmito systémy

lze též simulovat proudění živin (5). Kontextové L-systémy jsou odvozeny od kontextových gramatik. V mé podobě přepisovacích pravidel je kontext reprezentován spojitým stromovým grafem, v němž mohou proudit informace stejně jako živiny stromem. Uvádím příklad kontextového L-systému pro posílání živin (5)

$$Uloz = \{N, T, P, S\} \tag{5}$$
$$N = \{S, X, C\} \tag{6}$$
$$T = \{v, k\} \tag{7}$$
$$P = \{$$

$$S \to kvvvB, \tag{8a}$$
$$vC \to XC, \tag{8b}$$
$$XC \to Xv, \tag{8c}$$
$$Xv \to Cv, \tag{8d}$$
$$kC \to kk \tag{8e}$$

$$\}$$

Proces přepisování:

$$\text{Počáteční symbol je } S \tag{9}$$
$$\text{Aplikace pravidla (8a)} \Rightarrow kvvvC \tag{10}$$
$$\text{Aplikace pravidel (8b), (8c), (8d)} \Rightarrow kvvCv \tag{11}$$
$$\text{Aplikace pravidel (8b), (8c), (8d)} \Rightarrow kvCvv \tag{12}$$
$$\text{Aplikace pravidel (8b), (8c), (8d)} \Rightarrow kCvvv \tag{13}$$
$$\text{Aplikace pravidla (8e)} \Rightarrow kkvvv \tag{14}$$

Kontextová gramatika *Uloz* je tvořena těmito částmi. Terminály T, kde v představuje větev a k zastupuje kořen. Neterminály N, kde S je počáteční symbol, C představuje molekulu cukru, X je pomocný neterminál. Množina P obsahuje jednotlivá přepisovací pravidla. Gramatika *Uloz* simuluje část procesu ukládání cukrů do kořenů rostliny. Cukry jsou tvořeny fotosyntézou v listech. Následně vstupují do větví, aby se mohly uložit v kořenech. Pro jednoduchost budu uvažovat, že v rostlině byla vytvořena pouze jedna molekula cukru C. Ta se nyní chystá vstoupit do jedné z větví v. Cestou ke kořenu musí molekula cukru projít přes tři větve. Tuto situaci znázorňuje počáteční stav $kvvvC$, tři větve mezi cukrem a kořenem. Provedu tedy přepis (refeq:prepis1). Pomocí pravidel (8b), (8c), (8d) je cukr C schopný překonat jednu větev v směrem ke kořenu k. Když cukr doputoval (11) přes jednu větev , nachází se celý systém ve stavu $kvvCv$. Cukr se opakováním trojce pravidel stále přibližuje ke kořenu (12). Jakmile se dostane do kořene (13), je cukr spolu s půdními látkami zpracován. Což vede k růstu masy kořenového systému. Růst symbolizuje zvýšení počtu kořenů na dva (14). Přidáním další gramatiky pro získávání a posílání živin z půdy do větví, vznikne rostlina, která bere živiny z okolí, z nichž staví své tělo. Jak tělo virtuálních rostlin tvoří člověk se dozvíte v další kapitole.

2.3 Ruční modelování

Když počítače začaly pracovat s 3D grafikou. Objevily se také první filmy a 3D hry využívající modelů pro vizuální efekty. Samozřejmě se v nich objevovaly i rostliny. Výpočetní výkon nedovoloval používat pro aplikace běžící v reálném čase složitých modelů. Proto se rostliny nahrazovaly plochou texturou s obrázkem rostliny (bilboarding). Později byly modelovány stromy. Nebylo vhodné se zabývat modely menších rostlin, protože by prodlužovali výpočet obrazu scény. Model stromu se skládal z kmene, tvořeného přibližně osmi stěnami a místo koruny se umístilo několik zkřížených ploch s texturou listů (Obr. 6). Vývoj počítačů se však nezastavil a brzy dovolil běžným uživatelům zobrazovat složitějších scén v reálném čase. To vedlo k návratu L-systémů.

Obrázek 6: Ručně tvořené modely stromů

2.4 Návrat k L-systémům

Ručně vytvářené modely přetrvaly zhruba do roku 2005. V té době dovolilo zvýšení výkonu běžných osobních počítačů vytvářet složitější objekty. Ruční modelování detailněji zpracované vegetace bylo časově náročné. A tak se L-systémy vrátily, ale tentokrát již sloužily grafikům. Místo toho, aby definovaly chování jednotlivých buněk, byly pomocí L-systémů definovány zákonitosti větvení a růstu nových částí rostlin. Nejmenší jednotkou tedy nebyla buňka, ale části rostliny jako list větev či plod. Nejvíce užívaným řešením pro generování vegetace pomocí L-systémů je SpeedTree. Ten je stále používán pro většinu komerčních produktů v nichž se vyskytuje virtuální vegetace. Výsledné rostliny jsou opravdu pěkné.

Obrázek 7: Příklad L-systémem generovaných rostlin, zdroj [9]

Ovšem výkon běžných počítačů stále rostl. Zvětšovali se i zobrazované scény v nichž přibývalo rostlin. A grafici měli opět problém. Museli vkládat obrovské množství rostlin a

starat se o to, aby krajina vypadala věrohodně. Například skladba lesa musela odpovídala jeho nadmořské výšce, zdrojům živin a vody v krajině. Proto byl učiněn další krok, který vedl směrem k simulacím ekosystému.

2.5 Simulace ekosystému

Simulace prostředí nastoupila ve chvíli, kdy se v jedné scéně začalo objevovat velké množství travin, keřů a stromů. Ruční vkládání a rozmisťování vegetace na vhodná stanoviště stálo grafiky mnoho práce. Objevila se tedy poptávka po systému do, kterého jsou zadány parametry krajiny a zbytek už bude práce počítače. Parametry mohou zahrnovat informace o živinách v půdě, vlhkosti i konkrétní skladbě druhů. Systém sám, pak vytváří rostliny, které reagují jedna na druhou, obsazují si prostor a bojují o živiny a světlo. Systém se také sám stará o správnou aplikaci textur na modely. Prací grafika na rostlinách je jen upravit scény důležité pro příběh filmu či hry. K tomu dodá několik jedinců rostlin, kteří jsou něčí specifičtí a nelze je vygenerovat při simulaci. Simulace ekosystémů jsou aktuálně nejsofistikovanějším nástrojem pro vytváření virtuální vegetace pro větší scény [2].

Obrázek 8: Příklad simulace ekosystému, zdroj [9]

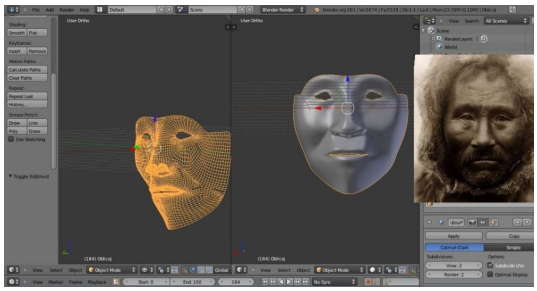
Do simulací začíná být začleňována práce s voxely, které usnadňují detekci kolizí a práci s informacemi v 3D prostoru. Voxely jsou rozmístěné do mřížky podobně jako u rastrového obrazu. S tím rozdílem, že další mřížky se přidávají směrem do výšky. Výsledný objekt vyplňuje 3D prostor. Voxely možná nejsou zase tak nový nápad (Obr. 6). Funkce voxelu se dá přirovnat k funkci rastru obrázku. Sám voxel neobsahuje svou polohu v prostoru, ale jeho účelem je udržovat informace. Ty udržují nejen data o barvě jako u obrázků, ale i o fyzikálních veličinách a dalších parametrech vztaženým k pozici voxelu. Zástupcem sekce simulátorů je opět SpeedTree, protože do sebe postupně včlenil i simulaci ekosystému. Zdroj informací o voxelech [4].

Obrázek 9: Příklad staršího využití voxelového přístupu v herním průmyslu

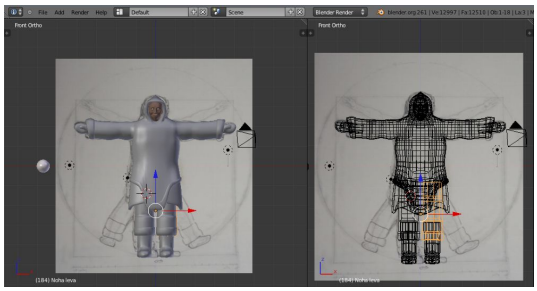
2.6 Nástroj Blender

Blender je open-source software požívaný pro modelování a vykreslování 3D počítačové grafiky a animací. Jeho největší výhodou je, že za ním stojí široká komunita, v které se nachází jak odborníci na práci s grafikou, tak i úplní amatéři, kteří se teprve učí pracovat s nástroji pro modelování a animace. Blender má za sebou dlouhý vývoj a s každou novou verzí přibývá mnoho rozšíření.

Tento komplexní nástroj dovoluje vytvářet a rozpohybovat modely, které si uživatel vytvoří. Nanášet na ně textury včetně normálových map. Při tom může využívat fyzikálního modelu, který může být použit pro práci s částicemi a simulace kapalin. V Blenderu lze také klíčovat speciální efekty do videa. Všechny tyto služby jsou nabízeny zdarma. Samozřejmě je možné se zapojit do vývoje systému. I když tyto nástroje nejsou placené vyrovnají se těm komerčním a v něčem je i překonají. Přikládám ilustrační obrázky z tvorby jednoho z mých projektů (Obr. 10)

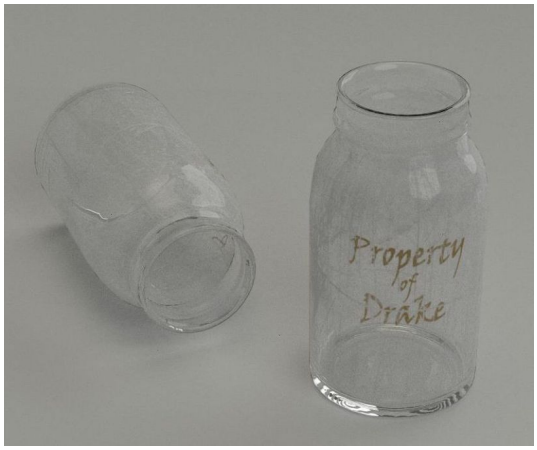


(a) Tvorba obličeje



(b) Tvorba postavy

(c) Dokončená postava



(d) Vytvořené sklenice

Obrázek 10: Příklad práce na některých mých projektech v nástroji Blender

2.7 Modul Sapling

Modul Sapling slouží pro vkládání parametrizovaných stromů do scény. Modul je v současné době vložen do nástroje Blender jako jeho stálá komponenta. Stromy vkládané do scény jsou tvořeny pomocí křivek. Modul jsem testoval jako uživatel, ale jeho ovládání pro mne nebylo pohodlné, především především kvůli velkému množství parametrů, které musí uživatel nastavit. Pro detailnější nastavení druhu mohou být užitečné. Bohužel hlavní položky, které jsou často měněny, jsou roztroušeny po více záložkách. Navíc se mi i přes veškerou snahu nepodařilo vnést do kmene stromu mírná zakřivení, aby vypadal reálněji. Došel jsem tedy k názoru, že autor nechává tuto práci na uživateli. Z toho důvodu je strom tvořen pomocí křivek. Ovšem je poté otázkou, proč má modul takové množství parametrů.

Listy, jsou vkládány na větve tak, že vypadají přirozeně, ovšem problém nastal u mapování textur. Autor používá standardně pro každý list hexagon, který je složen z šesti trojúhelníku. Na hexagon nedokáže Blender automaticky namapovat běžnou texturu listu. Naštěstí lze použít i čtverce. U nich nástroj zvládá automatické rozbalení UV-mapy čtvercových polygonů pro listy. Dalším problémem byla špatně nastavená integritní omezení. Modul směle dovolí rozmístit na větev -25 listů, při jejichž výpočtu přestane nástroj Blender reagovat a bez jediného slova se se mnou rozloučí. Tento modul je vhodnější pro tvorbu stromů, které jsou svou stavbou podobné smrkům či pěstovaných stromů. Ty mají mnoho rovných částí a proto už není nutná ruční úprava křivek kmenu a větších větví. Na jejich tvorbu je modul optimální.

Obrázek 11: Příklad využití modulu Sapling

2.8 IvyGen

Ivy Generátor se požívá pro vytvoření popínavých rostlin, které sledují tvar vybraného objektu. Protože jeho zdrojový kód je volně přístupný v modulech nástroje Blender, mohl jsem prozkoumat jeho princip. Je založený na L-systému, který je navíc schopen zjišťovat kolizi s objektem. Jeho pravidla jsou poměrně jednoduchá. V každém kroku nechá vyrůst šlahoun o jeden dílek. Dalším pravidlem je, že s danou pravděpodobností, kterou volí uživatel, vyraší z hlavního šlahounu jeden vedlejší. Pro vedlejší šlahoun platí opět pravidla jako pro hlavní. Jak již bylo řečeno, procesu růstu vstupuje ještě kontrola kolize. Pokud je v místě kam šlahoun poroste již jiný objekt, nebo pokud by se šlahoun od objektu příliš vzdaloval, je upraven směr jeho růstu tak, aby s určitou mírou tolerance sledoval tvar obrůstaného objektu. Nakonec může být nad každým dílkem vygenerována čtvercová ploška, reprezentující list. Pravděpodobnost, že list vyroste, kontroluje opět uživatel. Konec růstu nastane, když rostlina dosáhne délky zadané uživatelem, nebo je překročen zadaný maximální čas, pro vygenerování rostliny.

Obrázek 12: Využití modulu IvyGen v nástroji Blender

Modul IvyGen jsem vyzkoušel i jako uživatel (Obr. 12). Pracuje se s ním pohodlně a rychle. Výsledek vypadá věrohodně a správně kopíruje tvar obrůstaného objektu. Tělo rostliny je tvořeno křivkami, které jsou obtaženy válci. Oproti tomu jeden list odpovídá jednomu čtvercovému polygonu. Ve volbě textur listů a materiálu těla rostliny nechává volný výběr uživateli. Vytvořit textury pro čtvercové polygony listů není v Blenderu složité. Lze je vytvořit pomocí node editoru, se samostatných textur pro list a průhlednost. Nebo lze ručně přiřadit texturu s listem, která již obsahuje alfa kanál. Více o Ivy generátoru se lze dočíst zde [8].

2.9 SpeedTree

SpeedTree je softwarový nástroj poskytující komplexní řešení pro vytváření scén obsahujících virtuální vegetaci. Tento nástroj existuje již od roku 2003 a od té doby prošel značnými změnami. Nejprve byl zaměřen na vytváření jednotlivých stromů za pomoci L-systémů. Během let došlo k značnému rozšíření SpeedTree po celé komerční scéně. Výsledkem je rozdělení vývoje do dvou proudů. Prvním je tvorba scén pro realtime aplikace, jako jsou počítačové hry (Obr. 13a). A druhým je tvorba scén pro filmy a grafiku (Obr. 13b). K tomuto rozdělení došlo zřejmě, kvůli tvorbě filmu Avatar. Při níž tvůrci SpeedTree a autoři filmu vytvářeli sofistikovanější nástroje pro scény s důrazem na detailní zpracování vegetace. Bohužel se do zdrojových kódů SpeedTree lze podívat pouze v případě, že si jej koupíte a i tak veškeré změny patří tvůrcům SpeedTree. Ale je možné odzkoušet trial-verze zdarma jako běžný uživatel s omezením na dobu užívání dvaceti dnů.

Odzkoušel jsem několik verzí programu. Starší verze 4.0, která poskytovala nástroje pro modelovaní jednotlivých rostlin. Ovládání bylo přehledné i přesto, že šlo nastavit velké množství atributů pro každou generaci z částí rostliny. Později jsem měl možnost vyzkoušet verzi 6.0, v které již bylo vytváření rostlin rozšířeno. Do modelů se automaticky vkládala kostra. Program používal svůj vlastní engine pro simulaci fyziky. Díky kostře se celá rostlina mohla pohybovat ve větru. Při zatížení se ohýbaly větve a při překročení jejich nosnosti se lámaly. Bylo jasné že celý software se stále vyvíjí, aby vyhovoval náročnějším požadavkům zákazníků. Zdroj [9].

Nástroj Blender sice nemá tak dobře placený tým odborníků, ale zato má širokou základnu samostatných nadšených vývojářů a věřím, že jednou se také dostane na úroveň tvorby vegetace, kterou nabízí SpeedTree nyní. Snad bude prvním maličkým krůčkem i mnou vytvořený modul VegGen.

(a) SpeedTree for Games

(b) SpeedTree Cinema Studio

Obrázek 13: Využití SpeedTree, zdroj [9]

3 Modul VegGen

Hlavní částí této práce je návrh a implementace modulu pro nástroj Blender (Sekce 2.6). Mnou vytvořený modul umožňuje uživateli do scény vkládat modely stromů a keřů. Upravovat je změnou různých parametrů algoritmu růstu. Důraz jsem kladl na to, aby byl modul, co nejvíce nezávislý na knihovnách Blender. Jeho jádro se bude brzy měnit a já bych si rád ušetřil práci při budoucích úpravách pro udržení kompatibility. Další výhodou je, že Blender je multiplatformní.

Jádrem modulu je algoritmus umožňující aplikovat na rostlinu přepisovací pravidla. Nyní popíší hlavní kroky vytváření rostliny:

1. získání parametrů pro nastavení rostliny od uživatele
2. vytvoření přepisovacích pravidel
3. vytvoření axiomu (semínko rostliny)
4. vyvolání růstu o uživatelem zadaný počet iterací
5. aplikace přepisovacích pravidel se stochastickými změnami na axiom
6. příprava dat pro generování modelu v Blenderu
7. předání dat do Blenderu
8. zobrazení

Při tvorbě modulu jsem dodržoval pravidla vícevrstvé architektury, protože plánuji jeho jádro v budoucnu použít pro jinou samostatnou aplikaci. Modul lze tedy logicky rozdělit do 4 vrstev. První vrstvou jsou nositelé dat. Tvoří ji spoje, větve, přepisovací pravidla a axiom. Spoje a větve modul využívá pro vnitřní reprezentaci rostliny. Obsahují tedy informace o poloze i vlastnostech částí rostliny. Oproti tomu přepisovací pravidla a axiom využívá vyšší vrstva simulátoru růstu pro proces vytváření nových částí rosliny na základě těch starších.

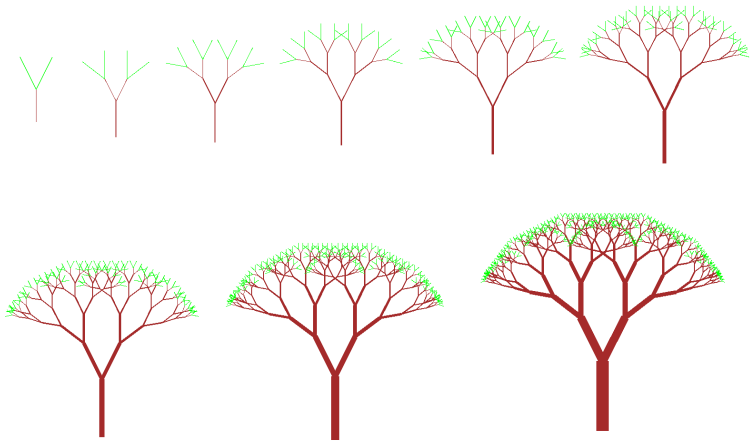
Druhou vrstvou je Simulátor růstu. Je jádrem modulu, jelikož umožňuje aplikovat na rostlinu přepisovací pravidla a vytvářet složitou strukturu rosliny. Lze v něm kombinovat více pravidel, čímž lze například na větvích nechat vyrůst plody nebo používat pro různě staré části rostliny různé tvary. Prozatím však vrstva, která se stará o polygonizaci, generuje pouze větve a listy.

Třetí vrstva provádí převod z interních dat na polygony. Vrstvu tvoří Vizualizátor, který dostane jako vstupní data stromovou strukturu vygenerovanou simulátorem růstu. Z těchto dat je schopen Vizualizátor vytvořit body. Následně mezi nimi vygenerovat stěny. Podle toho zda se uživatel rozhodne pro optimalizovaný model tvořený trojúhelníky a čtyřúhelníky, a nebo model tvořený pouze čtyřúhelníky. V budoucnu chci místo generování těla rostliny složeného z polygonů vytvářet model pomocí křivek. Připravená data zobrazí až čtvrtá vrstva.

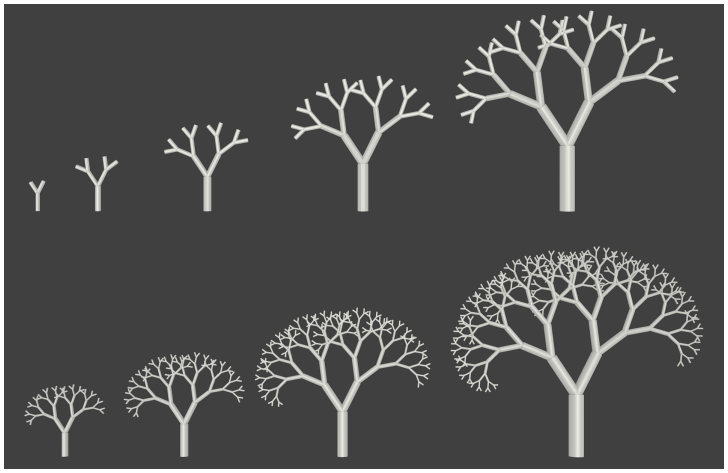
Čtvrtá vrstva slouží jako řídící člen a také komunikuje s prostředím nástroje Blender. V této vrstvě definuji konkrétní přepisovací pravidla a přesměrovávám výstupy a vstupy vrstev tak, abych je spojil ve funkční celek. Jednoduše řečeno, čtvrtá vrstva umí použít přírodní síly a vizualizaci výsledků k získání výsledného modelu tvořeného polygony. Více se o funkcích vrstev dozvíte níže.

3.1 Data a jejich struktura

Mým prvním pokusem byl prototyp programu, který vytvářel obrázky stromů v dvourozměrném prostoru. Z toho důvodu mi pro ukládání struktury rostliny stačila halda přímek, které jsou na sobe funkčně nezávislé. Protože výsledky prototypu (Obr. 14) byly uspokojivé, provedl jsem také první pokusy s vložením rostliny do prostředí nástroje Blender (Obr. 15). Prozatím stále v jedné rovině za pomoci válců.



Obrázek 14: Výsledky prototypu na plátně, 1 až 9 generace

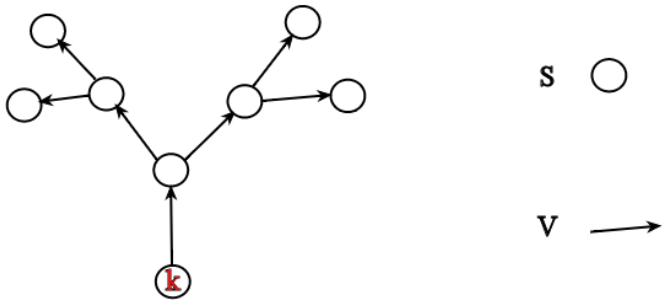


Obrázek 15: Výsledky prototypu v nástroji Blender, 1 až 9 generace

Ovšem nedostatky této struktury se projevily v okamžiku, kdy jsem se pokusil vnést do růstu rostliny vliv náhody a toho, zda větev roste směrem vzhůru nebo je spíše převislá. Nespojitý model v těchto případech nedovoloval zjistit, které větve na sebe navazují, a které se již dále nevětví. I kdybych se pokusil zjistit spojitosti mezi koncovými a počátečními body přímek porovnáním polohy, nebyla by zaručena správnost výsledku. Protože není řešena kolize větví, může ve stejném bodě začínat nebo končit více oddělených částí rostliny. Hlavním problémem ovšem bylo, jak ze separovaných přímek vytvořit spojitý objekt. Řešení jsem objevil ve stromových strukturách.

3.1.1 Spoje a větve

Z důvodu nedostatků, které má datová struktura tvořená oddělenými přímkami, jsem se pokusil vytvořit optimálnější model. Výsledkem mého zkoumání je stromová struktura. Pro názornost uvádím vzorový nákres této struktury (Obr. 16).



Obrázek 16: Koncept datového modelu

Struktura obsahuje uzly S neboli spoje a větve V. Hlavní úlohou uzlu je nést informaci o poloze v prostoru. Každý uzel zakončuje jednu větev. Výjimkou je uzel k, jelikož je kořenový. Zároveň v uzlu může začínat libovolný počet větví nových. Větev je orientovaná spojnice mezi dvěma uzly a její úlohou je nést informace o části rostliny. Stromová struktura odstraňuje problémy haldy přímek. Umožňuje procházet rostlinu, jak směrem od kořene k listům a naopak. Jednoduše v této struktuře zjistím vazby jedné větve na sousední, včetně jejich vlastností. Pokud změním polohu bodu, ze kterého vyrůstá více větví, aplikuje se transformace na všechny navázané větve. Nyní lze vytvořit výsledný model rostliny jako jeden spojitý objekt.

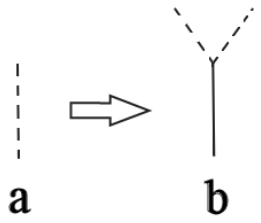
Stromová struktura přináší i další výhody. Díky tomu, že jsem odstranil duplicitní nosiče informace o poloze, jsem ušetřil paměťové prostředky. Navíc se snížila časová náročnost výpočtů. Kupříkladu kdybych chtěl přesunout tentýž tvar jako na (Obr. 16) tvořený oddělenými přímkami, musel bych přepočítat polohu čtrnácti bodů. Ve stromové struktuře se jedná jen o přepočet osmi bodů. Jediné co by mohlo být považováno za nevýhodu, je možnost vzniku cyklů při nesprávné definici tvaru. Tato chyba však nemůže vzniknout samovolně, při vytváření rostliny programem, ale pouze špatnou definicí pravidel, či axiomu. Z čehož vyplývá, že chybu lze odstranit kontrolou vstupních dat, tedy vyhledáním cyklů v grafech, před provedením samotného procesu tvorby rostliny. Prozatím však může tyto tvary zadávat pouze programátor, od kterého se očekává jistá technická zdatnost.

3.1.2 Přepisovací pravidla

Přepisovací pravidlo popisuje pomocí uzlů a větví, jakým tvarem nahradit či rozšířit starou část rostliny, když chci získat novou iteraci. Přepisovací pravidlo tedy musí obsahovat tvar, který chci nahradit, i tvar, který umístím na jeho místo. Nejlepší bude uvést příklad takového pravidla (Obr. 17), jehož textovým zápisem je (2).

Tvar a je původní část rostliny. Tvar b je, nová iterace, která starou část nahradí. Jak je patrné nová generace obsahuje dvě úsečky a, což zajišťuje možnost růstu v dalších iteracích. Pro generátor, který bude generovat rostlinu je tedy nutné implementovat přepis pravidel. Nejprve modul VegGen umožňoval používat pouze jedno pravidlo, které bylo tvořeno právě původní úsečkou a novým tvarem. Pozice původní úsečky a nového tvaru

musí korespondovat, kvůli následujícím rotacím a změnám při procesu vytváření nových iteracích. Pro ulehčení následných výpočtů umisťuji původní úsečku na osu z a její počátek je v počátku souřadného systému. eq:zavorky



Obrázek 17: Příklad přepisovacího pravidla

Užívání jediného pravidla značně omezovalo množství i rozmanitost rostlin, které šli vytvořit v rámci jednoho druhu. Proto jsem vytvořil entitu *Pravidlo* reprezentující přepisovací pravidlo a jeho pojmenovaní. A zároveň jsem u větvě rozšířil vlastnost, informující zda bude větev přepisována, o to kterým pojmenovaným pravidlem se má přepsat. Díky tomu lze vytvářet rostlinu více pravidly. Což umožňuje nechat na rostlině vyrůst plody. Jako další vylepšení entita *Pravidlo* ve skutečnosti nese soubor *1 až n* jednotlivých vážených pravidel, ze kterých se náhodně, s přihlédnutím na váhu pravidla, vybírá vždy jedno, které se použije. Čím vyšší má pravidlo váhu, tím je větší pravděpodobnost, že bude vybráno. Díky tomuto vylepšení, lze například vytvořit *Pravidlo KMEN*. To generuje kmen, který má šanci $1 : 8$, že se bude (váha 1) nebo nebude (váha 8) větvit. Nyní už zbývá jen nadefinovat počáteční útvar. Na nějž budu aplikovat pravidla. Tento útvar se nazývá axiom.

3.1.3 Axiom

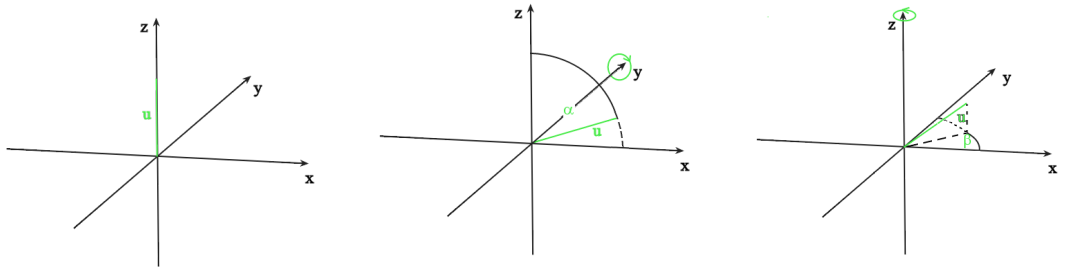
Axiom je v podstatě semínko nebo také prvotní útvar, ze kterého vyroste celá rostlina. Pro vytvoření axiomu využívám též stromovou strukturu (Sekce 3.1.1) tvořenou spoji a větvemi. Například pro strom je axiomem jednoduchý tvar tvořený dvěma spoji, provázanými jednou větví. Ta nese informaci o tom, že bude přepisována pravidlem pro tvorbu kmene. Spoj, v kterém axiom začíná, je kořenovým spojem celé budoucí rostliny. Pro uložení tvaru axiomu si stačí zapamatovat tento kořenový spoj. Díky vlastnostem stromové struktury jsem schopen projít všechny jeho části. Nyní můžu aplikovat na semínko pravidla růstu, tedy přepisovací pravidla k získání rostliny určitého stáří a velikosti. K tomuto účelu jsem vytvořil Simulátor růstu.

3.2 Simulátor růstu

Vstupními daty pro Simulátor růstu jsou: přepisovací pravidla, axiom, stochastické proměnné a požadovaná generace. Samotný princip simulátoru růstu není ve své podstatě složitý. Tato vrstva vytváří strukturu rostliny. Růst začíná axiomem. V každé další iteraci se na každou větev aplikují přepisovací pravidla. Postup práce simulátoru by mohl vypadat jako na obrázku růstu závorkového L-systému 5. Ale během tohoto procesu se též uplatňuje stochastický přístup. Ten provádí změny tvarů. Avšak jen v uživatelem stanoveném rozmezí. Realizace nahrazení staré části rostliny novou není tak prosté. Nejprve musím vědět, jak je vůbec stará část natočena v prostoru.

3.2.1 Natočení části rostliny v prostoru

Jako výchozí natočení jsem zvolil rovnoběžné s osou z (Obr. 18a). Natočení staré části určuji jako posloupnost otočení o úhel α kolem osy y (Obr. 18b), a po té o úhel β kolem osy z (Obr. 18c). Nyní jen otočím pořadí kroků, abych získal hodnoty úhlů α, β. Nejprve vezmu větev u kdekoliv v prostoru a otočím ji kolem osy z, aby náležela do roviny xz. Úhel tohoto otočení je úhel β. S větví v rovině xz snadno zjistím úhel α. Důležité je pouze dávat si pozor na správnou orientaci úhlů. Nyní mohu vzít nový útvar z řídícího pravidla



(a) Pozice na ose z (b) Rotace kolem osy y (c) Rotace kolem osy z

Obrázek 18: Natočení větve v prostoru

a natočit jej stejně jako původní větev. Postačí projít všechny spoje, a rotovat jimi kolem počátku řídící větve pravidla. Abych nemusel provádět jednotlivé rotace kolem os, převedu souřadný systém do homogenního prostoru a sestavím si jednu transformační matici, kterou vynásobím všechny spoje. Převedení do homogenního prostoru je jednoduchá operace, v níž přiřadím všem bodům stejnou homogenní souřadnici w, volím rovno 1.

3.2.2 Transformační matice

$$T_p = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, T_\gamma = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, T_\beta = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$
$$T_\alpha = \begin{pmatrix} \cos\gamma & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, T_s = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, T = T_p T_\gamma T_\beta T_\alpha T_s \tag{15}$$

Výsledná matice T provádí tyto operace v uvedeném pořadí:

1. maticí T_s provedu změnu měřítka, kde s_x, s_y, s_z jsou jednotlivé změny v osách x, y, z
2. maticí T_α provedu rotaci kolem osy z o úhel α; rotace větve kolem své osy
3. maticí T_β provedu rotaci kolem osy y o úhel β; natočení do polohy jako stará větev
4. maticí T_γ provedu rotaci kolem osy z o úhel γ; natočení do polohy jako stará větev
5. maticí T_p provedu posun na pozici původního tvaru o souřadnicích (t_x, t_y, t_z)

Homogenní transformační matice umožňují skládat jednotlivé rotace, posuny, a změny měřítka do jediné matice [1]. Což je výhodné právě v případech, kdy chci nad množinou bodů provádět stejné operace, protože všechny výpočty se zredukují na operaci násobení vektoru a matice. Při násobení matic jednotlivých operací je třeba dávat pozor na to, že matice se násobí v obráceném pořadí, oproti posloupnosti operací prováděných s body. Výše jsem popsal ‚jak získám výslednou transformační matici T (15).

3.2.3 Proces přepisu větve

Tento proces je klíčový pro vygenerování struktury celé rostliny. K získání každé nové generace projdu celou stávající strukturu rostliny. Pro každou větev zjistím, zda je v ní nastaven symbol jednoho z přepisovacích pravidel (Sekce 3.1.2). Kdyby větev uschla nebo neobsahovala žádný přepisovací symbol, má práce s ní končí. Jinak pokračuji dále a vyhledám si na základě přepisovací symbolu patřičný útvar, který se stane novou generací této větve. Než však budu moci nahradit stávající větev, musím nový útvar řádně upravit.

Nejprve zjistím natočení současné větve v prostoru (Sekce 3.2.1) o úhel α kolem osy y a o úhel β kolem osy z. Také musím vědět, kolikrát zvětšit nebo zmenšit nový tvar, aby velikostí odpovídal současné větvi. K tomu stačí porovnat délku současné větve a délku původního části rostliny v přepisovacím pravidle nového útvaru. Výsledný poměr je hodnota změny měřítka s pro nový útvar na všech osách. Pro simulaci vlivu prostředí obměním s pro každou osu v intervalu zadaným uživatelem. Tak vzniknou změny měřítka $s_x,\ s_y,\ s_z$. Ve skutečně rostlině nerostou vlákna většinou rovnoběžně, ale tvoří spirálu. Proto také každý útvar nové generace otáčím kolem jeho osy o jeden krok, úhel δ. Díky toho, že se tato rotace se provádí před ostatními, nový útvar opravdu rotuje kolem své osy, kterou je osa z. Naopak poslední úpravou je celý nový útvar posunout tak, aby se jeho počátek, který je teď shodný s počátkem souřadného systému, nalézal na stejné pozici jako počátek přepisované větve. To znamená, že hodnota posunu je rovna poloze počátečního spoje větve $[\ t_x,\ t_y,\ t_z\ ]$. Mám všechny potřebné informace pro sestavení transformační matice T (Sekce 3.2.2).

$$\vec{v'} = T\vec{v} \tag{16}$$

Jako další krok projdu všechny spoje nového útvaru. Z polohy bodu vytvořím vektor $\vec{v}$, rozšířený o homogenní souřadnici $w = 1$ a transformační matici T vynásobím tímto vektorem (16). Novou polohou spoje je výsledný vektor $\vec{v'}$, převedený zpět do afinního prostoru. Nyní má nový útvar správnou pozici, natočení i velikost. Posledním krokem je odstranit stávající větev s jejím koncovým spojem. Její počáteční spoj dosadím jako počáteční spoj nového útvaru. Takto pokračuji pro všechny větve rostliny v rámci růstu o jednu generaci. Během procesu přepisování je třeba dávat pozor, aby do procesu nevstoupily nově přidané části rostliny. Tento problém jsem odstranil kontrolou iteračního kroku, v němž větev vytvořena. Pro získání další generace opakuji tento postup. Uvedený proces již obsahuje některé vlivy náhody, které činí strukturu rostliny více podobnou rostlinám reálným. Celý proces tvorby rostliny, však ovlivňuje stochastický přístup daleko více, než je uvedeno výše.

3.2.4 Stochastický přístup

Protože modul VegGen generuje samostatné stromy bez složité simulace ekosystému, nahradil jsem elementem náhody působení vnějších vlivů. Jinou možností, jak řešit vlivy

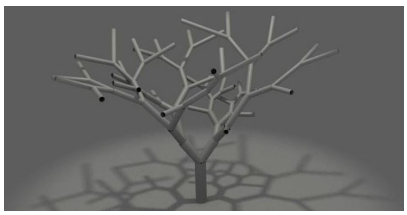
prostředí na rostlinu, by bylo nechat uživatele zadávat tyto proměnné. Ze začátku jsem si myslel, že by to mohl být i dobrý nápad. Jenže problém se objeví v okamžiku, kdy před program posadíte běžného uživatele a pro vložení jediného objektu je třeba nastavit 100 parametrů. Jen málokomu by se chtělo pouštět do sáhodlouhého zadávání hodnot. Pokud by už někdo chtěl tvořit rostliny konkrétních tvarů, sáhne dozajista po programech, které jsou sofistikovanější a dovolují pár pohyby myši vytvořit model rostliny pomocí křivek, které se sami obalí libovolnými tvary a texturami. Zaměřil jsem se proto jiným směrem. Podíval jsem se na požadavky běžného uživatele:

- vygenerování reálně vypadající rostliny
- možnost nastavení druh (smrk, lípa, bez, šípek atd.)
- určení staří, vzrostlá
- nastavení míru detailů
- možnost vložení více rostlin stejného druhu, různých tvarů
- minimální úsilí pro vytvoření různých rostlin podobných vlastností

Běžný uživatel tedy chce do své scény snadno a rychle umístit rostliny a detailní tvarování každé větvičky ho obtěžuje. Z toho důvodu, je modul nastaven tak, že jde ovlivnit velikosti intervalů, v kterých se bude náhodná hodnota pohybovat. Stochastický přístup je uplatněn pro výběr přepisovacích pravidel, na změnu tvaru a velikosti nově vyrůstajících částí rostliny i na rostlinu jako celek. Po rychlém nastavení druhu rostliny lze jednotlivé exempláře snadno generovat změnou inicializačního čísla generátoru pseudonáhodných číslic. Což představuje nastavení pouze jediné položky.

3.3 Polygonizace

Výsledným produktem simulátoru růstu je struktura rostliny tvořená spoji a větvemi. Další krok je vytvořit na základě této struktury tvary rostliny, které budu moci ukázat uživateli. Já jsem zvolil reprezentaci tělesa pomocí hranice. Budu jej skládat z polygonů, tedy trojúhelníkových a čtvercových polygonů. Tento způsob jsem zvolil, protože jsem se chtěl naučit tvořit taková tělesa pomocí algoritmů v nástroji Blender. Otázkou tedy



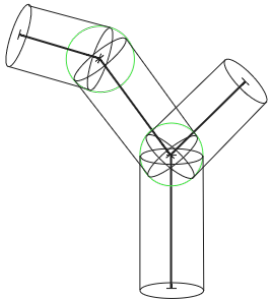
Obrázek 19: Nespojené válce

zůstává jak, převést stromovou strukturu větví a spojů na trojrozměrnou geometrii rostliny. Nejednodušším postupem by bylo na základě polohy větve a číselném údaji o poloměru vytvořit válec složený z polygonů. Jak je vidět z Obrázku 19 toto těleso nevypadá jako rostlina. Což způsobuje ,že válce na sebe nejsou navázané a tudíž je vidět dovnitř tělesa. Tyto nespojitosti by šlo vyřešit tak, že na místa spojů umístím klouby ve tvaru

koule. Dále jsem čerpal inspiraci z práce [6]. Pro vykreslování autor užívá želví grafiku, kterou já nepoužívám. Je zde uvedeno několik obecných principů, jak tvořit geometrii. Princip tvořící čistější topologii sám autor nezkoušel implementovat. Ovšem já jsem tvorbu čistější topologie implementoval a vylepšil (Sekce 3.3.2).

3.3.1 Geometrie válců s klouby

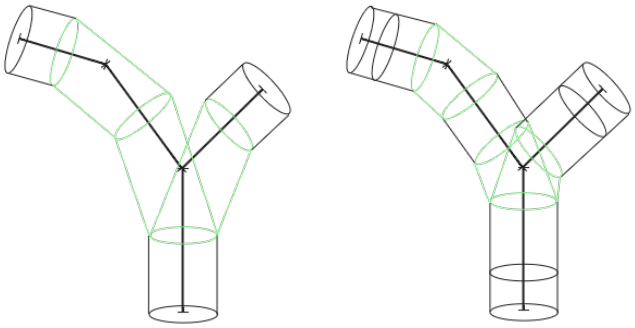
Provázání sousedních válců pomocí kulových kloubů, je jednoduchým řešením. Místo každé větve vygeneruji válec. Mezi konce válců, které na sebe mají navazovat a tudíž mají stejný poloměr podstav, umístím kulový kloub. Kloub má poloměrem stejným jako má podstava válců. Tímto způsobem se překryje nespojitost (Obr. 20). Tento postup však přináší některé nevýhody. Hlavním je zbytečně vysoký počet polygonů, které přibývají s každým kloubem. Také se mohou při určitých úhlech osvětlení vyskytnou, problémy objevujících se stínů, způsobených přiblížením ploch kloubu a válce. Topologicky správným řešením je tedy vytvořit geometrii rostliny ze spojitých válců.



Obrázek 20: Geometrie válců s klouby

3.3.2 Geometrie spojitých válců

Při tvorbě této geometrie nevznikají válce jako obraz každé větve. Místo nich vznikají z každé větve pouze body rozmístěné na kružnici. Válce vznikají následným propojováním sousedních kružnic polygony (Obr. 21a).



(a) Spojitá geometrie

(b) Vylepšená spojitá geometrie

Obrázek 21: Spojité válce

Během tohoto procesu tedy nekopíruje výsledná topologie přesně tvar stromové struktury. Což je cenou za to, že získávám spojitou geometrii. S touto nepěknou deformací jsem se nechtěl smířit. Takže jsem provedl malé vylepšení, aby byla výsledná geometrie více podobná původní struktuře reprezentující větev. Vytvořil jsem místo jedné kružnice dvě, které jsou umístěny ve třech čtvrtinách a v jedné čtvrtině délky větve (Obr. 21b).Díky tomu se ve výsledném objektu objevuje válec odpovídající části původní větve z datové struktury. Zároveň se tvoří propojovací válec zmenšující úhel, pod kterým přechází jedna větev v další. Celá rostlina díky tomu vypadá přirozeněji.

Při tvorbě geometrie je také možné volit mezi pevným počtem polygonů, který bude pro všechny válce stejný, nebo nastavit pouze úroveň detailů, kterou algoritmus použije spolu s poloměrem větve k výpočtu množství bodů tvořících spojové kružnice. Při nastavení úrovně detailů budou tlustější větve vytvořeny z více polygonu než větve tenké. Nastavení je tedy vhodné pokud se ve scéně budu sledovat na celou rostlinu. Nebudu sledovat pouze detail její drobné části.

Aby bylo možné vytvářet optimalizovaný model, implementoval jsem vlastní algoritmus pro propojovaní kružnic s rozdílným počtem bodů. V prvním kroku definuji proměnné:

```
faces = deque()          # array like [], seznam polygonů
top = vIndxs1    #mnozina vrcholu uzke krurnice
bot = vIndxs2    #mnozina vrcholu sirsi kruznice
B = rozdilVertexuNa1top = Fraction(len(bot)-len(top),len(top))
topIdx = 0 #aktuane vybrany bod z uzke krurnice
botIdx = 0 #aktuane vybrany bod z sirsi krurnice
```

Kvůli úspoře počtu polygonů je vykresleno jen nejmenší potřebné množství trojúhelníků. Zbytek válce je tvořen čtverci. Proto je třeba stanovit koeficient $rozdilVertexuNa1top$. Samotný rozdíl počtu bodů kružnic udává minimální počet trojúhelníků potřebných pro vytvoření souvislé plochy spolu s čtverci. Koeficient vztahuje rozdíl na jeden dílek kružnice s menším poloměrem. Dále uvedu princip rozhodování:

```
pokracuj = True
while pokracuj:
    while B >= 1 and botIdx < len(bot) and topIdx < len(top)-1:
    #vykresli trojuhelnik
        B -= 1
        face = [ bot[botIdx], top[0], bot[botIdx + 1] ]
        faces.append(face)
        botIdx += 1
        if (botIdx >= len(bot)-1) and (topIdx >= (len(top)-1)):
            pokracuj = False
    while B < 1 and botIdx < len(bot)-1 and topIdx < len(top)-1:
    #vykresli obdelnik
        B += rozdilVertexuNa1top
        face = [bot[botIdx], top[topIdx],
                top[topIdx + 1], bot[botIdx + 1]]
        faces.append(face)
        topIdx += 1
        botIdx += 1
        if botIdx >= len(bot)-1 and topIdx >= len(top)-1:
            pokracuj = False
```

Na začátku je hodnota B rovna $rozdilVertexuNa1top$. S každým nakresleným obdélníkem se však hodnota B zvýší. To odráží rostoucí vzájemné posunutí množin bodů kružnic (dále jen kružnic) *top* a *bot*. Tento posun je způsoben tím, že z obou kružnic odebírám na každý čtverec 1 bod. Protože kružnice *bot* je širší zbývá jí propojit více bodů než kružnici *top*. Z toho důvodu musím na vhodných místech vykreslit trojúhelník. Tento svou základnou vezme bod z kružnice *bot*, ale svým vrcholem se pouze napojí na bod z kružnice *top*. Neodebere tak možnost použít jej při dalším vykreslování čtverců. Pokud kreslím trojúhelník, vyrovnávám posunutí aktuálně propojovaných bodů kružnic. Proto se hodnota B snižuje o 1. Na kreslení trojúhelníků přejdu když hodnota $B >= 1$. Na kreslení čtverců se vracím když $B < 1$. Tento princip lze využít pro vytvoření plochy. Pokud je však požadováno kreslení válců, je nutné propojit čtvercovým polygonem první a poslední body kružnic. Jak ukazuje kód níže.

```
if cyclic:
    face = [ top[0],bot[0], bot[len(bot)-1],top[len(top)-1]]
    faces.append(face)
```

Ať už vytvářím válce s pevným nebo zmenšujícím se počtem bodů je výsledkem množina vrcholů, a množina stěn, z kterých ve vyšší vrstvě vytvořím objekt pro nástroj Blender.

3.3.3 Generování listů pro keře a stromy

Poslední věcí, která ještě chybí mým dřevinám jsou listy. Jejich vytvoření je už poměrně snadné. Jako listy mi poslouží samostatné čtvercové polygony. Takže z jejich vytvořením není problém. Problémem je jen jak a kam je umístit. Pokud se podívám na listy vyrůstající ze stromů a keřů, rostou vždy na nejmladších větvích. Přičemž jsou většinou rozestavěny po délce větve ve spirále s pravidelným krokem (Obr. 22). Krok může být i π. To způsobí, že listy jsou všechny v jedné rovině s větví. Další obměnou je dvojitá spirála, kde listy rosou na proti sobě. Postup vytvoření listů v modulu VegGen (Obr. 23) je následující. Určím

(a) Pupeny (b) Listy (c) Jehlice

Obrázek 22: Příklady rozložen zeleně na větvích

pozice jednotlivých listů a jejich natočení. Nejdříve je rozmístím tak, jako by na ose Z byla umístěna jejich větev s počátkem v bodu $[0, 0, 0]$. Sestavím transformační matici, kterou bych větev přesunul z její aktuální pozice na pozici, kde se skutečně nalézá. Každé čtyři vrcholy polygonu listu převedu do homogenního prostoru a vynásobím touto maticí. To znamená že se listy rozestaví na svá konečná místa. Z výsledných pozic sestavím množinu vrcholů a množinu stěn všech listů. Množina bodů a množina stěn jsou výsledkem tohoto procesu.

Obrázek 23: Výsledné rozložení listů v modulu VegGen

3.4 Vrstva pro komunikaci s nástrojem Blender

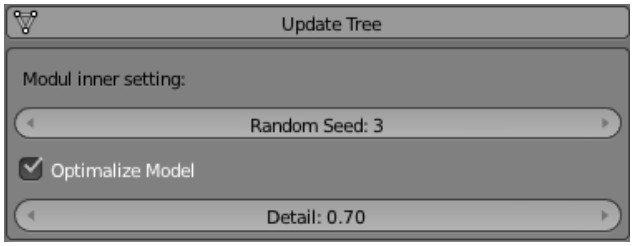
Vrstva pro komunikaci spojuje hned několik funkcí. Řídí práci nižších vrstev. Obsahuje pokyny pro Blender, jak vytvořit grafické rozhraní pro uživatele. Získává data od uživatele. Vytváří z množin vrcholů a stěn objekty, které předává do prostředí Blender. Obsahuje rozhraní, díky kterému Blender umí s tímto modulem zacházet. Toto rozhraní a způsob jak s ním pracovat popisují manuálové stránky nástroje Blender [7]. I když můj způsob jak se naučit rozhraní vytvořit a pochopit byl nastudovat funkčnost již hotových modulů, protože Blender je konec konců open-source nástroj.

3.4.1 Uživatelské rozhraní a vstupní data

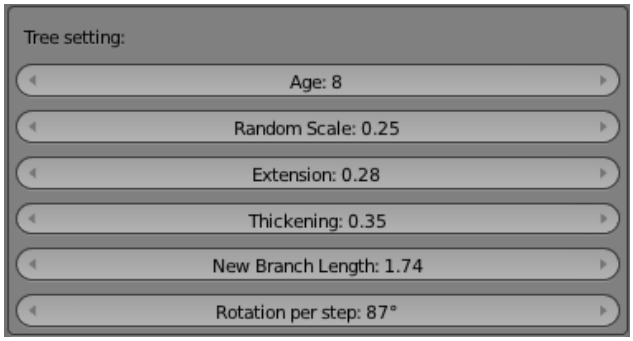
Aby mohl modul reagovat na požadavky uživatele, musím mu dovolit měnit parametry procesu při, kterém se vytváří rostlina. Proto je důležité vytvořit uživatelské rozhraní. To (Obr. 25) jsem pro lepší přehlednost rozdělil na vnitřní nastavení generátoru (Obr. 25a). Nastavení mající význam v rámci celého stromu a měnící jeho topologii (Obr. 25b). A nakonec nastavení samotných větví, které je spjato s nastavením listů(Obr. 25c). Nejdůležitější hodnotou, kterou bych chtěl zmínit, je inicializační hodnota pro generátor náhodných čísel *Random Seed* v nastavení generátoru. Díky ní lze rychle generovat různé jedince se stejně nastaveným rostlinným druhem.

Obrázek 24: Nastavování parametrů rostliny nástroji Blender

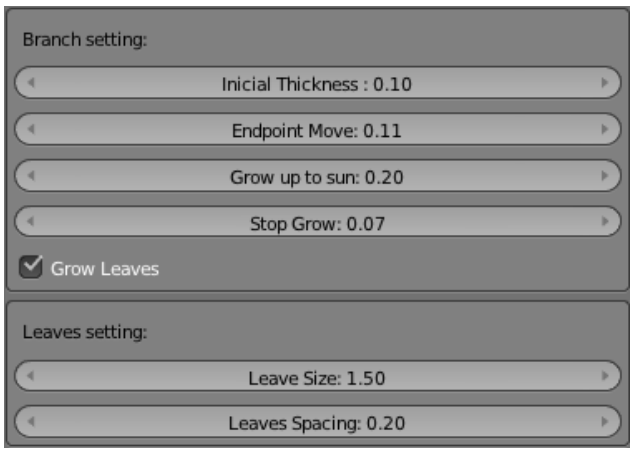
V prostředí Blenderu je tvorba uživatelské rozhraní velice jednoduchá. Díky předdefinovaným proměnným, které v sobě mají zabudovanou i vizualizaci pro uživatelské rozhraní. Jejich hodnota je při změně uživatelem automaticky aktualizována. Při tomto procesu je volána funkce, která vygeneruj rostlinu z aktuálních hodnot proměnných a touto nahradí starou verzi. Okamžité překreslení však není pro modul VegGen vhodné. Vygenerování složitějších exemplářů rostlin může trvat i více než deset vteřin, což by bylo pro uživatele při nastavování proměnných nepohodlné. Proto jsem aktualizaci geometrie rostliny svázal s tlačítkem *Update* (Obr. 25a), a uživatel si tak může nejprve nastavit všechny parametry bez zbytečného čekání. Až poté aktualizovat geometrii. Když je nyní jasné, jak získám data od uživatele, nadešel konečně čas spustit modul. Více se o procesu tvorby rostliny dozvíte v další kapitole.



(a) Vnitřní nastavení generátoru



(b) Nastavení stromu



(c) Nastavení větví

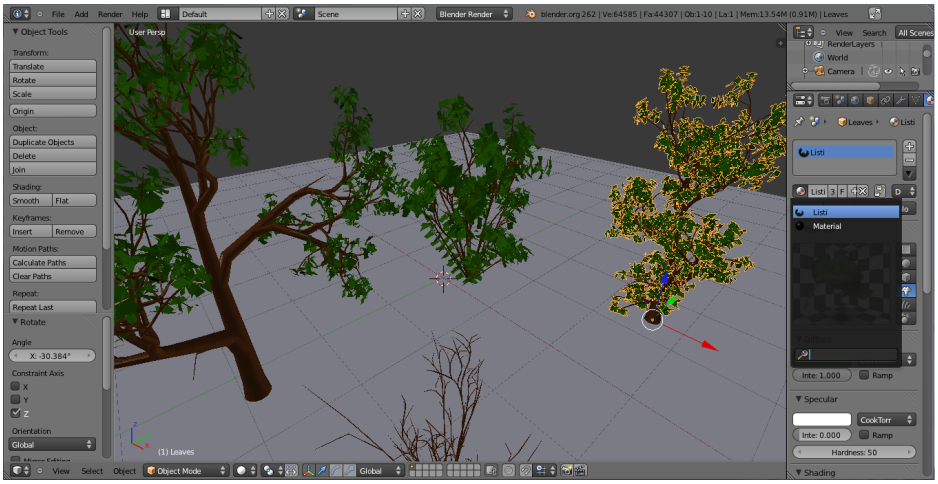
Obrázek 25: Uživatelské rozhraní

3.4.2 Řídící funkce

Řídící funkcí je myšleno, že vrstva využívá a spravuje všechny nižší vrstvy, aby mohla vytvořit výsledný objekt v prostředí nástroje Blender. K tomu potřebuje nadefinovat axiom i přepisovací pravidla. Pro strom vytvořím například pravidlo růstu kmene, z kterého v každém patře vyroste zároveň nová větev. Musím tedy nadefinovat i pravidla pro růst větví. Při tvorbě pravidel jíž zohledňuji některé ze vstupních parametrů zadaných uživatelem. Z těchto parametrů nejvíce ovlivňuje tvar pravidel poměr délky nových a starých větví.

Dalším krokem je použít simulátor růstu s přepisovacími pravidly na axiom, semínko, a nechat jej vyrůst o počet iterací zadaný uživatelem. Během růstu se samozřejmě uplatňují náhodné změny některých parametrů. Ovšem pouze v takových rozmezích, jaké zadal uživatel. Výsledkem procesu růstu axiomu je stromový graf tvořený větvemi a spoji. Obsahuje informace potřebné pro vytvoření výsledného modelu rostliny. Čtvrtá vrstva vezme výsledek růstu a předá jej vrstvě, která strukturu umí převést na množinu vrcholů a stěn budoucích polygonů. Každý vrchol je v množině reprezentován svými souřadnicemi. V množině stěn je každá stěna reprezentována trojicí nebo čtveřicí čísel. Tato čísla jsou pořadovým číslem vrcholu v množině vrcholů. Představují tedy body, které jsou spojeny, aby vytvořili jednu stěnu. Pokud uživatel zadal, že chce nechat rostlinu obrůst listím, vygeneruji pro ně také množiny vrcholů listů a stěn listů.

Když čtvrtá vrstva obdrží výsledné množiny, vytvoří kontejner objektu pro nástroj Blender. Umístí do něj množiny vrcholů a stěn. Nechá kontejner, aby si z dat vytvořil polygony. Kontejner se po té předá správci objektů nástroje Blender. Pokud se generují listy, vytvoří stejný kontejner i pro ně. Do kontejneru opět vložím data a nechám vytvořit polygony. Následně kontejner listů přiřadí jako položku kontejneru těla rostliny. Tím je proces dokončen a objekt rostliny se objeví v pracovním prostředí nástroje Blender. Uživatel si může rostlinu upravovat. Přidávat materiály a textury jejím listům i jejímu tělu a vůbec s ní může pracovat jako s každým jiným objektem (Obr. 26). Nyní už zbývá jen modul řádně odzkoušet, čemuž se věnuji v další kapitole.



Obrázek 26: Práce s výsledným modelem v nástroji Blender

4 Experimenty

V této kapitole se zabývám porovnáváním výsledku modulu VegGen se skutečnými stromy. Modul není primárně určen pro tvorbu rostlin v nichž, chce uživatel nastavit přesný tvar každé větve a listu. Přesto jsem se pokusil najít pro vybrané listnaté stromy a keře z reálného světa odpovídající protějšky v mém modulu. Listnaté stromy a keře jsem vybral, protože pravidla, která zatím používám pro generování rostlin, nezahrnují jehličnany. V budoucnu chci vypracovat podrobnější pravidla pro více druhů rostlin, z kterých si bude uživatel moci vybírat.

Předtím než jsem začal generovat jednotlivé stromy připravil jsem si nástroj Blender. Vytvořil jsem několik materiálů pro kůru a půdu a jeden materiál pro listy. Umístil jsem scénu a nasvítil ji. Následně jsem skrze uživatelské nastavení nástroje Blender nainstaloval a aktivoval modul VegGen. Stejně jako se přidávají i ostatní nenainstalované moduly. Stromy a keře je po aktivaci modulu možné vkládat z menu *Add Mesh* ≫ *Vegetation*.

4.1 Tvorba stromů

Postup

Vložil jsem model stromu a nastavil jeho druh pomocí parametrů ve skupinách *Tree Setting* a *Branch Setting*. Skrze změnu hodnoty *Random Seed* jsem generoval různé jedince vytvořeného druhu stromu. Nejdéle po deseti minutách jsem našel model velice podobný předloze. Vždy jsem provedl úpravy rotace větví, aby jejich rozestavení více připomínalo vzor. Jen vzorové stromy experimentu 4 (Obr. 30b) bojovali o světlo s ostatními rostlinami. Z toho důvodu u jejich větví převládal vertikální směr. Proto musela být navíc v modelech (Obr. 30a) zvýšena hodnota parametru *Grow Up*. Ten určuje míru protažení větví směrem vzhůru. Jako poslední jsem přidal na větve listy a nastavil jejich velikost a hustotu, pokud se listy vyskytovaly i na vzoru. Tímto končila práce modulu VegGen.

Celý model rostliny jsem zmenšil a mírně sklonil dle předloh a natočil vhodnou stranou ke kameře, v případě že toho bylo třeba. Vybral jsem z předem připravených materiálů vyhovující kůru stromu stejně jako materiál pro listy. Pokud nebylo pozadí a barva kůry modelu dostatečně kontrastní, přiřadil jsem pozadí materiál s vhodnou barvou půdy. Následovalo spuštění procesu *Render* a čekání na výsledky.

Výsledky

V této části jsou pro porovnání uvedeny vzorové fotografie a výsledné snímky vytvořených modelů.

(a) Předloha

(b) Výsledek

Obrázek 27: Experiment 1

(a) Předloha

(b) Výsledek

Obrázek 28: Experiment 2

(a) Předloha

(b) Výsledek

Obrázek 29: Experiment 3

(a) Předloha

(b) Výsledek

Obrázek 30: Experiment 4

4.2 Tvorba keřů

Postup

Vložil jsem model keře a stanovil poměrné zvětšení délky větví za jeden cyklus parametrem *Extension* a poměr délky mladých větví ke starých *New Branch Length*. Protože vzory neměli složitou strukturu větví, nastavil jsem stupeň iterace $Age = 4$. Pro tvorbu druhu keře nebylo potřeba dalších změn oproti počátečnímu nastavení. Dalším krokem bylo generování různých jedinců keřů volbou *Random Seed*. Výsledný keř jsem ovšem musel sestavit vždy z dvou keřů stejného druhu postavených vedle sebe. Důvodem bylo rozdělení nadzemních částí vzorových rostlin do dvou skupin. Toto je způsobeno růstem keřů pomocí oddenků podzemních částí rostliny. Po vytvoření modelů končí práce modulu VegGen.

Následují změny provedené nástrojem Blender. Jelikož u vzorů se bod, z kterého větve vyrůstají, nenachází v nadzemní části rostliny. Musel jsem i u svých keřů tento bod umístit pod plochu tvořící povrch půdy. Celý model rostliny byl zmenšen a skloněn dle předloh. Poté byl natočen vhodnou stranou ke kameře. Protože vzorové keře se více natahovali do výšky provedl jsem navíc roztažení ve vertikálním směru. Zbývalo pouze z předem připravených materiálů zvolit vyhovující kůru. Vzorové keře byly bez listí, tudíž toto nebylo potřeba přidávat. Pokud nebyl povrch zeminy a barva kůry modelu dostatečně kontrastní, přiřadil jsem pozadí materiál s vhodnou barvou půdy. Posledním krokem bylo spuštění procesu *Render* a čekání na výsledné snímky.

Výsledky

(a) Předloha (b) Výsledek

Obrázek 31: Experiment 5

(a) Předloha (b) Výsledek

Obrázek 32: Experiment 6

4.3 Zhodnocení

Při tvorbě stromů pracoval modul uspokojivě. Uživatelské rozhraní pro mne bylo příjemné. Opravdu praktické je pro mne umístění volby parametru *Random Seed* v blízkosti tlačítka *Update*. Z důvodu časté změny tohoto parametru a následného spuštění přepočtu geometrie. Ačkoliv není modul určen k vytváření rostlin přesných tvarů podařilo se mi vygenerovat stromy dle předlohy. A to v kratším čase než když jsem je kdysi tvořil ručně. Také manuální přidávání listů dotýkajících se stonkem větve by bylo pracné.

Tvořit keře se mi dařilo během několika minut. Jejich podobnost s vzory však není tak velká jako u stromů. Úpravou přepisovacích pravidel bych mohl dosáhnout kvalitnějších výsledků. Keře však ve většině případů nebývají hlavní součástí scény. Pokud se navíc nechají porůst listy, nejsou nedokonalosti patrné. Protože vytvořené keře nemají mnoho polygonů pomáhají k vyplnění prostoru scény s malým dopadem na celkový čas potřebný k vytvoření snímku.

Nejlépe však čtenář zhodnotí provedené experimenty pokud sám porovná vygenerované rostliny a vzorové.

5 Závěr

Během tvorby mé bakalářské práce jsem získal mnoho zajímavých poznatků a především cenné zkušenosti. Na počátku jsem nastudoval problematiku tvorby virtuálních rostlin a její historii i využití v současnosti. Díky tomu jsem značně rozšířil své povědomí o současných trendech ve vytváření virtuálních scén. Pronikl jsem hlouběji do principů tvorby fraktálních těles a procedurálních modelů.

Získané znalosti jsem využil pro vytvoření generátoru virtuálních stromů a keřů. Díky volbě provázat můj projekt s nástrojem Blender ve formě modulu VegGen se možnosti práce s mým generátor značně rozšířily. Uživatel snadněji upraví výsledné rostliny tak, aby vyhovovaly jeho scéně. Může na ně aplikovat nepřeberné množství nástrojů či dalších modulů, které nabízí prostředí aplikace Blender.

S výsledkem své práce jsem spokojen a dokonce předčila má původní očekávání od tohoto projektu. Stále však vidím nevyužitý potenciál modulu. Chtěl bych jej proto nadále rozvíjet a podporovat. Nejprve chci, aby rostlina nebyla vykreslována polygony ale pomocí křivek. Díky tomu bude pro uživatele snazší mapovat textury pro kůru a také upravovat tvar již vygenerované rostliny. Taktéž chci vypracovat podrobnější pravidla pro více druhů rostlin, z kterých si bude uživatel moci vybírat. V budoucích verzích chci propojit modul s částicovým systémem Blenderu. Na pozici každé z částic by se generovala jedna rostlina. Tímto způsobem by bylo možné snadno pokrýt vybrané plochy vegetací. Což uživateli značně urychlí práci s vytvářením scény. Jakmile se mi podaří tato vylepšení dokončit, nabídnu svůj modul vývojářům nástroje Blender a jeho komunitě. Získám tak zpětnou odezvu a možná i velké množství nových nápadů pro budoucí vývoj.

Vedle práce na modulu samotném bych jeho jádro chtěl využít pro vytvoření samostatné aplikace pro mobilní zařízení. Tato aplikace by fungovala jako simulace pěstování malé rostlinky na pracovní ploše zařízení. Uživatel by mohl zastřihávat listy i části rostliny. Samozřejmě by musel rostlinu zalévat a přihnojovat. Pro tyto účely bych vylepšil stávající systém o simulaci proudění živin. Toto mi snadno umožní implementovat mnou navržená stromová struktura (Sekce 3.1.1), kterou využívám pro interní reprezentaci rostliny.

Celkově pro mne byla má bakalářská práce dobrodružnou cestou za sebepoznáním. Musel jsem při ní překonávat mnoho překážek i své vlastní pochyby. Nakonec se mi však podařilo dorazit do cíle a nyní s nadějí zaměřuji svůj zrak do dálek a hledám nové výpravy.

Reference

[1] Eduard Sojka, Martin Němec, Tomáš Fabián, *"Matematické základy počítačové grafiky"* [online] c2011, Dokument dostupný z: `http://mi21.vsb.cz/modul/matematicke-zaklady-pocitacove-grafiky` [citováno 2.březen 2012]

[2] Hans Pretzsch, *"Forest Dynamics, Growth and Yield: From Measurement to Model"*, Berlin: Springer, 2009

[3] Benoît Mandelbrot, *"Fractals and Chaos"*, Berlin: Springer, 2004

[4] Lengyel, Eric. *"Transition Cells for Dynamic Multiresolution Marching Cubes"* Journal of Graphics, GPU, and Game Tools. Vol. 15, No. 2 (2010), A K Peters

[5] Aristid Lindenmayer *"Mathematical models for cellular interaction in development* Díl I a II, Journal of Theoretical Biology, 1968.

[6] Marek Pasičnyk, *"Generování rostlin pomocí L-systémů"*, Brno: Masarykova univerzita, 2011

[7] Blender python API, *"Blender 2.5 Python 3.2 Manual"* [online] `http://wiki.blender.org/index.php/Doc:2.6/Manual/Extensions/Python`

[8] Ivy generator, *"Ivy Generator"* [online] `http://graphics.uni-konstanz.de/~luft/ivy_generator/`

[9] SpeedTree, *"SpeedTree Software"* [online] `http://www.speedtree.com/`

[10] Aplikace Google Earth, *"Google Earth"* [online] Aplikace je zdarma dostupná z: `http://www.google.com/earth/index.html`

Seznam příloh

- CD s kompletním obsahem práce a modulem VegGen